Authentication: Who can? Who can't?

Lesson Notes | Support | Complete & Continue

In this lesson you'll add authentication (so that a user can't delete another user's pins).

(https://github.com/onemonthrails/pinteresting/tree/7b109d497e56c829ae0a4e1a7bd1e654b99815b9) Browse Source Code (https://github.com/onemonthrails/pinteresting/tree/7b109d497e56c829ae0a4e1a7bd1e654b99815b9)

Update the Pins Controller

app/controllers/pins_conroller.rb

+

```
class PinsController < ApplicationController
  before_action :set_pin, only: [:show, :edit, :update, :destroy]

  def index
    @pins = Pin.all
  end

  def show
  end

  def new
    @pin = current_user.pins.build
  end

  def edit
  end

  def create
    @pin = current_user.pins.build(pin_params)
    if @pin.save
      redirect_to @pin, notice: 'Pin was successfully created.'
    else
      render action: 'new'
    end
  end

  def update
    if @pin.update(pin_params)
      redirect_to @pin, notice: 'Pin was successfully updated.'
    else
      render action: 'edit'
    end
  end

  def destroy
    @pin.destroy
    redirect_to pins_url
  end

  private
    # Use callbacks to share common setup or constraints between ac
tions.
    def set_pin
      @pin = Pin.find(params[:id])
    end

    def correct_user
      @pin = current_user.pins.find_by(id: params[:id])
      redirect_to pins_path, notice: "Not authorized to edit this p
in" if @pin.nil?
    end

    # Never trust parameters from the scary internet, only allow th
e white list through.
    def pin_params
      params.require(:pin).permit(:description, :image)
    end
end
              +
```

Update the pins view

app/views/pins/index.html.erb

```
<%= pin.user.email if pin.user %>
```

Or alternatively you could the Ruby "try" (http://api.rubyonrails.org/classes/Object.html) method....

```
<%= pin.user.try(:email) %>
```

(I don't choose this one, but it's good to know about)

Add devise User authentication

Resource: https://github.com/plataformatec/devise (https://github.com/plataformatec/devise)

Add the before_action to your Pins Controller

```
before_action :authenticate_user!, except: [:index, :show]
```

Surround the edit link with an "if" conditional

This way you can only see your pins. To put that another way: A user can only see his pins (and not other user's pins). Make sense?

app/views/pins/index.html.erb

```
...
<% if pin.user == current_user %>
  <%= link_to 'Edit', edit_pin_path(pin) %>
  <%= link_to 'Destroy', pin, method: :delete, data: { confirm: 'Ar
e you sure?' } %>
<% end %>
...
```

app/views/pins/show.html.erb

```
...
<% if @pin.user == current_user %>
  <%= link_to 'Edit', edit_pin_path(pin) %>
<% end %>
<%= link_to 'Back', pins_path %>
...
```

Add correct_user method

Add the before_action to your Pins Controller

```
before_action :correct_user, only: [:edit, :update, :destroy]
```

Surround the "New Pin" link with an "if" conditional

app/views/pins/index.html.erb

+

```
...
<% if user_signed_in? %>
  <%= link_to 'New Pin', new_pin_path %>
<% end %>
...
```

Q: Why do we sometimes use @ before variable names?

We have to use a `@` symbol before pins because we are pulling it from the controller. In a previous example in *app/views/pins/index.html.erb* we just used `pins.user` without the symbol because it was already assigned (in the loop above on line #15: `<% @pins.each do |pin| %>`)

+